(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 169 587 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
02.08.2017 Bulletin 2017/31

(21) Application number: **08017143.2**

(22) Date of filing: **29.09.2008**

(51) Int Cl.:
G06F 21/60 *(2013.01)* ***G06F 21/62*** *(2013.01)*

(54) **Method and rule-repository for generating security-definitions for heterogeneous systems**

Verfahren und Regelspeicher zur Erzeugung von Sicherheitsdefinitionen für heterogene Systeme

Procédé et règle de dépôt pour générer des définitions de sécurité pour des systèmes hétérogènes

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MT NL NO PL PT RO SE SI SK TR

(43) Date of publication of application:
31.03.2010 Bulletin 2010/13

(73) Proprietor: **Software AG**
64297 Darmstadt (DE)

(72) Inventors:
- **Weber, Heiko**
  64319 Pfungstadt (DE)
- **Harbarth, Juliane**
  64347 Griesheim (DE)

(74) Representative: **Heselberger, Johannes**
Bardehle Pagenberg Partnerschaft mbB
Patentanwälte, Rechtsanwälte
Prinzregentenplatz 7
81675 München (DE)

(56) References cited:
EP-A- 1 308 823
US-A1- 2006 010 445
US-B1- 7 308 702
WO-A-2008/103725
US-B1- 6 978 379

EP 2 169 587 B1

Printed by Jouve, 75001 PARIS (FR)

Description

1. Technical Field

[0001] The present invention relates to a method and a rule-repository for generating security-definitions for heterogeneous systems.

2. The Prior Art

[0002] Today, huge software systems providing a large amount of diverse data are typically combinations of different heterogeneous subsystems such as internet portals, collections of different databases or registries, which makes the overall systems very complex. Heterogeneous in this context means that the subsystems each may have a completely different structure, underlying technology, data formats, processing concepts, etc..

[0003] In order to ensure the security of the data in such a composed software system, e.g. to ensure that only authorized parties have access to parts of the data or that only authorized personnel is allowed to perform certain system tasks, the subsystems typically each allow for the declaration of a set of security definitions. However, the heterogeneous subsystems might specify fine-grained security definitions by different technical means and conforming to different syntactic models. On the other hand, the overall security of the software system is likely to be the result of coarse-grained, global security intents, which have to be manually manifested by administration personnel in the multiple different fine-grained security definitions of each subsystem. Since a global security intend may likely affect multiple different subsystems, the various different security definitions have to be kept synchronized at all times in order to guarantee the security of the overall software system. Thus, even to a knowledgeable administrator, the overall set of a system's fine-grained security definitions may often appear as an incomprehensible collection of access privileges, which is difficult to maintain and results in the risk of security leaks of the overall software system.

[0004] In this context, the US 7,058,715 describes how to manage access control within system topologies, in particular in storage-area networks. It discloses a formal mathematical model to group accessors and accessibles into proto-zones in order to facilitate the security management. Furthermore, the US 6,751,509 discloses a method to achieve access control to tabular data organized in cells and aggregations of cells, especially in closed systems such as financial institutions. The document focuses on the derivation of aggregated access control rules on the same fine-grained abstraction-level.

[0005] US 6,978,379 B1 relates to an apparatus for use in generating configuration information for a computer system employing hierarchical entities. Disclosed are definitions shown by a user interface to a user, policy templates and refined policies, as well as a bidirectional mapping between the definitions and the policy templates. The starting point of the technique disclosed in US 6,978,379 B1 are the templates which are manually created by an expert. Starting from them and from a model of the IT environment a wizard generates a natural-language description for the user, an internal representation of the templates and further assists the user in creating the refined policies.

[0006] Further methods for creating and executing security policies are disclosed in EP 1 308 823 A2 and WO 2008/103725 A1.

[0007] It is therefore the technical problem underlying the present invention to facilitate the derivation of fine-grained security-definitions from more global security intents, thereby increasing the security of huge software systems comprised of heterogeneous subsystems and thereby at least partly overcoming the above explained disadvantages of the prior art.

3. Summary of the Invention

[0008] This problem is according to one aspect of the invention solved by a method for synchronizing one or more system-specific security-definitions for multiple heterogeneous subsystems of a software system to enforce the security of the software system. In the embodiment of claim 1, the method comprises among others the following steps:

a. storing, in a rule-repository, one or more natural language security-definitions in XML syntax reflecting a global security intend for the software system which affects the multiple heterogeneous subsystems;

b. storing, in the rule-repository, one or more machine-readable security-definitions defined in XACML and a first mapping of each natural language security-definition onto one or more of the machine-readable security-definitions; and

c. automatically generating the one or more system-specific security-definitions in XML syntax from the one or more machine-readable security-definitions by one or more rule-converters corresponding to the one or more heterogeneous subsystems, wherein the generating is based on a second mapping of each machine-readable security-

definition onto one or more of the system-specific security-definitions.

[0009] Accordingly, the embodiment defines a method which facilitates the generation of system-specific security-definitions specific to different heterogeneous subsystems of a software system. As explained in the introductory section, each subsystem might have its own method of establishing security and especially might have its own data model of storing security definitions, which makes it very difficult to achieve the overall security-intends of the overall system. To this end, the embodiment defines a three-layered architecture for security-definitions, including a meta-representation of the system-specific security-definitions:

- Since every security-definition added to the system (as well as every update or deletion of such a definition) has an intent, i.e. follows some kind of rule, these rules are first explicitly formulated in natural language and stored in a rule-repository.
- Secondly, the natural-language security-definitions are represented in a machine readable form in the rule-repository. These machine-readable security-definitions must describe the intent of the original natural-language definitions as precisely as possible and sufficiently generic. Furthermore, a mapping between those abstraction layers is provided. Preferably, the mapping maps one natural-language security-definition onto one machine-readable security-definition. However, other types of mappings are also possible, e.g. 1:m or even n:m mappings.
- Lastly, in order to technically enforce the formulated security intents on the subsystem layer, a special rule-converter is provided preferably for each subsystem, which generates the system-specific security-definitions from the higher layer information according to a second mapping. The second mapping is preferably a 1:n mapping, however, other types of mappings are also possible, e.g. 1:1 or n:m mappings. Furthermore, the second mapping is for each of the one or more heterogeneous subsystems preferably automatically generated by the respective rule-converter.

[0010] Consequently, storing the security-definitions according to this three-layered security-architecture already has advantages in documenting changes to the security set, thus making it better maintainable. Additionally, the three-layered security-architecture provides a better understanding of which security-definitions exist and their intents and further achieves a synchronized generation of the fine-grained system-specific security-definitions from the more abstract security-intents. Thus, the overall security of the software system is improved through the enforcement of the security intents by the different system-specific security-definitions.

[0011] The method of claim 1 further comprises the steps of storing one or more new system-specific security-definitions in the rule-repository and automatically generating one or more machine-readable security-definitions from the one or more new system-specific security-definitions by the one or more rule-converters according to the second mapping. Accordingly, the rule-converters may be able to convert the security-definitions in both directions. This enables a "reverse-engineering" of existing fine-grained security-definitions, which facilitates a better understanding of the security-definitions and makes the validation of existing configurations easier, thereby further increasing the overall security.

[0012] In another aspect, the method further comprises the steps of selecting at least one of the natural-language security-definitions and retrieving the one or more affected system-specific security-definitions according to the first and second mapping. A system-specific security-definition is affected by a natural-language security-definition in this context, if it was derived from it through one or more corresponding machine-readable security-definitions. In this aspect, the rule-repository may be thought of as a combined data basis of rules in layers 2 and 3, which is able to provide information of which actions have taken place with respect to security. This is especially advantageous when the security activities undertaken related to a natural-language security-definition do not have the desired effects and the system-specific security-definitions are of such quantity that they cannot be efficiently inspected manually. Additionally or alternatively, the method may comprise the further steps of selecting at least one of the system-specific security-definitions and retrieving the one or more affected natural-language security-definitions according to the first and second mapping.

[0013] Furthermore, the method may further comprise the steps of updating the one or more natural-language security-definitions and / or the one or more machine-readable security-definitions and the step of updating the corresponding system-specific security-definitions by the one or more rule-converters. This aspect enables the security-definitions at the different abstraction layers to be kept synchronized.

[0014] In addition to retrieving layer 3 definitions affected by layer 1 definitions as presented above, the method may further comprise the steps of updating one or more of the affected system-specific security-definitions, selecting the one or more affected system-specific security-definitions which have been updated, and retrieving the at least one natural-language security-definition which has updated the selected system-specific security definitions. Accordingly, this allows for a kind of "impact analysis" of new layer 1 definitions on existing layer 1 definitions.

[0015] In order to detect if a security-definition has been updated, the one or more natural-language security-definitions, the one or more machine-readable security-definitions and / or the one or more system-specific security-definitions may comprise a time-stamp indicating the time of their last update.

[0016] Furthermore, the one or more machine-readable security-definitions may comprise any of the group of constructs

comprising users, roles, resources and / or access-privileges. This allows for a sufficiently precise but still generic representation of the corresponding natural-language security-definitions.

[0017] According to the invention, the one or more machine-readable security-definitions are defined in XACML. The *extensible access control markup language* (XACML) is a standardized declarative access control policy language implemented in XML.

[0018] The present invention also relates to a rule-repository for generating one or more system-specific security-definitions for one or more heterogeneous subsystems of a software system, the rule-repository comprising one or more rule-converters, wherein the rule-repository is adapted for performing any of the methods presented above.

[0019] Further advantageous modifications of embodiments of the method and the rule-repository of the invention are defined in further dependent claims.

[0020] Lastly, the present invention concerns a computer program comprising instructions for performing any of the methods presented above.

4. Short Description of the Drawings

[0021] In the following detailed description, presently preferred embodiments of the invention are further described with reference to the following figures:

Fig. 1: A schematic view of a three-layered security-architecture according to an embodiment of the present invention;

Fig. 2: A concrete example of the security-architecture in the context of two subsystems and corresponding rule-converters;

Fig. 3: A code listing illustrating an exemplary natural-language security-definition and a mapping onto the layers 2 and 3;

Fig. 4: A code listing illustrating an exemplary machine-readable security-definition in XACML;

Fig. 5: A code listing illustrating an exemplary system-specific security-definition for a WebDAV repository; and

Figs. 6a-b: Code listings illustrating an exemplary system-specific security-definition for a Tamino XML database management system.

5. Detailed Description

[0022] In the following, a presently preferred embodiment of the invention is described with respect to a three-layered security-architecture as schematically shown in Fig. 1. As can be seen, the security-architecture comprises an uppermost layer L1 comprising one or more natural-language security-definitions 100, a second layer L2 comprising one or more machine-readable security-definitions 200 as well as a mapping between the layers L1 and L2 (illustrated by the arrow between the security-definitions 100 and 200). The architecture further comprises a third layer L3 comprising one or more system-specific security-definitions 310, 311, 320 and 321. In the example of Fig. 1, the security-definitions 310 and 311 are specific for a subsystem S1 and serve to control access to resources S10 and S11 of the subsystem S1. Accordingly, the security-definitions 320 and 321 are specific for a subsystem S2 and serve to control access to resources S20 and S21. The connection between the machine-readable security-definitions 200 and the system-specific security-definitions 310, 311, 320, 321 is facilitated by corresponding rule-converters RC1 and RC2, which can generate layer-3-definitions from layer-2-definitions and vice-versa. The rule-converters RC1, RC2 are preferably each adapted specifically for the respective subsystem S1, S2.

[0023] Fig. 1 only shows an extremely simplified scenario. It should be understood that the present invention also relates to much more complex scenarios, possibly comprising hundreds of heterogeneous subsystems each with hundreds of security-definitions.

[0024] Furthermore, it may be appreciated that the system-specific security-definitions 310, 311, 320, 321 may not only serve for access control to resources S10, S11, S20, S21, but to fulfill any kind of security intent. For example, the system-specific security-definitions 310, 311, 320, 321 may define rights to perform certain tasks such as user-management, administration, etc. or rights to start certain system-specific processes such as monthly billing, bank transfers, mail delivery, etc..

[0025] Typically, there is no one-to-one mapping of the security-definitions on the different layers, since a higher-layered security-definition might well result in multiple security-definitions on a lower layer. To be able to convert in both

directions - from higher to lower and from lower to higher layer, an association between the security-definitions and their corresponding security-definitions on the other layers is given by the mappings depicted by the white arrows in Fig. 1. Maintaining these associations bears the advantage that they can be used for analytical purposes, as described further below.

[0026] In the following, various advantageous features of the present invention are illustrated in the context of the exemplary scenario in Fig. 2, where a company stores project-related information in a WebDAV repository S1 and in an XML database management system (DBMS) S2. The WebDAV repository S1 supports WebDAV access control lists (ACLs) 310, 311 according to the internet standard RFC3744 and the XML DBMS S2 supports Tamino's structure-based ACLs 320, 321 of Applicant. As a representation for the machine-readable security-definitions 200, the above described XACML is used.

[0027] The overall security-intent, i.e. the natural-language security-definition 100 in this example is to "grant read access to all developers on the data relating to the project SYLT".

[0028] The data relating to the project SYLT are stored in the WebDAV repository S1 folder "http://webdavserver/projects/SYLT/" and in the XML DBMS S2 collection "SYLT". The developers are all members of the group "Developers" which is uniquely identified by the URL "http://webdavserver/groups/Developers" in the WebDAV repository S1 and by the name "DevelopersGroup" in the XML DBMS S2.

[0029] In the following, all information is stored in a rule-repository in XML syntax. It should, however, be appreciated that XML is only one of a wide variety of possible data formats. Also, the specific structure of the shown XML documents is not intended to limit the scope of the invention to this structure. Furthermore, the rule-repository may be an XML database such as Tamino of Applicant. However, it should be appreciated that any other kind of storage facility is suitable for the present invention.

[0030] Fig. 3 shows how the natural-language security-definition 100 introduced above is embedded in an XML element *nlrule.* Furthermore, the mapping of this natural-language security-definition 100 onto corresponding security-definitions on the layers L2 and L3 is incorporated into the XML document of Fig. 3. Depending on the chosen structures for storing the representations of the different layers, the mapping information may also be maintained in a separate set of mapping documents. In other embodiments, where the rule-repository may be a Tamino XML database, the information of the three layers may be stored in one collection comprising a doctype "nrule" for storing the natural-language security-definitions 100, a doctype "policy" for storing the machine-readable security-definitions 200 and a specific doctype for each type of subsystem S1, S2 of the software system. However, any other type of storage structure may be suitable for the present invention.

[0031] The corresponding machine-readable security-definition 200 is shown in Fig. 4, which in this example is formulated in XACML. As can be seen, an XACML rule generally follows the form to define a certain *effect* (cf. Fig. 4, line 3) of *actions* (cf. Fig. 4, lines 21-28) performed by *subjects* (cf. Fig. 4, lines 5-12) onto *resources* (cf. Fig. 4, lines 13-20). Accordingly, the XML document in Fig. 4 directly reflects the natural-language security-definition 100 ("grant read access to all developers on the data relating to the project SYLT"), in that it defines, in XACML syntax, a rule with the effect to *permit* (cf. Fig. 4, line 3) *developers* (cf. Fig. 4, line 8) to *read* (cf. Fig. 4, line 24) the resource *SYLT* (cf. Fig. 4, line 16). The mapping between these rules is defined in Fig. 3, line 7, in that a *PolicyId*-attribute of the machine-readable security-definition 200 is referenced by the natural-language security-definition 100.

[0032] The corresponding WebDAV ACL 310, 311 for the WebDAV repository S1 generated from the machine-readable security-definition 200 from Fig. 4 by the corresponding rule-converter RC1 is shown in Fig. 5. As can be seen, the rule-converter RC1 has generated the XML document in Fig. 5 so that it conforms to the WebDAV-specific format. It has replaced the layer-L2-subject *developers* by the URL "http://webdavserver/groups/Developers" in a *D:principal*-element and has inserted a *D:privilege*-element <*D:read*>, which corresponds to the layer-L2-action *read.* The layer-L2-resource SYLT is implicitly defined in this layer-L3-definition, since the definition is directly applied on the WebDAV folder "http://webdavserver/projects/SYLT/".

[0033] Similarly, Fig. 6a shows a structure-based ACL 320, 321 for the Tamino XML DBMS S2 generated by the rule-converter RC2. As can be seen, this ACL binds the resource *SYLT* to the access type *read* (cf. Fig. 6a, line 3). In Tamino, such a structure-based ACL is bound to the respective subject by a group-definition as depicted in Fig. 6b. As can be seen, the ACL (referenced by its name *developers-readsylt*) is bound to the group *DevelopersGroup.*

[0034] Accordingly, the Figs. 5, 6a and 6b illustrate how the rule-converters RC1, RC2 can generate system-specific security-definitions 310, 311, 320, 321 from machine-readable security-definitions 200. It should be appreciated that although the above examples only show how the constructs of the machine-readable security-definitions 200 may be converted into constructs of WebDAV ACLs and Tamino structure-based ACLs, the present invention is not limited to these two specific data formats, but that any other rule-converter for other formats may operate in the present invention.

[0035] Since the information of all layers is represented in XML in the above example, XQueries may be formulated to analyze the information in order to research for security issues. It should be understood that any format other than XML and related query mechanisms may be employed in the present invention. The following sample queries assume that the security-definitions of all layers are collectively stored in a single collection "rules", however, other storage

structures are equally possible.

[0036] The following query retrieves all layer-L3 security-definitions that are affected by the natural-language security-definition with the Id "DeveloperReadRule", according to the mappings:

```
declare namespace D="DAV:"
declare namespace ino="..."
let $rules := collection("rules")
let $nlrule in $rules/nlrule[@ruleId =
    'DevelopersReadRule']
    let $layer3rules :=
   for $mapping in $nlrule/layermapping/layer3mappings
   return if ($mapping/@type = "WebDAV-acl")
        then $rules/D:acl[@id = $mapping/@id]
        else if ($mapping/@type = "Tamino-acl")
            then $rules/ino:acl[ino:aclname =
                $mapping/@id]
            else $rules/ino:group[ino:groupname =
                $mapping/@id]
                return $layer3rules
```

[0037] The next query depicted below takes this set and selects those layer-L3 security-definitions that have been touched, i.e. updated, in the meantime. This preferably involves a function *getTimestamp()* that computes the time-stamp a rule has been last updated. The function is preferably implemented in a way that heeds the different ways the time-stamp may be represented in the system-specific security-definitions of the different subsystems (e.g. in that it always returns a value of type *xs:datetime*).

```
declare namespace D="DAV:"
declare namespace ino="..."
let $rules := collection("rules")
let $nlrule in $rules/nlrule[@ruleId =
   "DevelopersReadRule"]
   let $layer3rules := ...
   let $touchedLayer3rules
   [getTimestamp(.) > getTimestamp($nlrule)]
   return $touchedLayer3rules
```

[0038] The last query takes this set and selects those natural-language security-definitions which caused the layer-L3 security-definitions to be updated:

```
declare namespace D="DAV:"
declare namespace ino="..."
let $rules := collection("rules")
let $nlrule in $rules/nlrule[@ruleId =
   "DevelopersReadRule"]
   let $touchedLayer3rules
   [getTimestamp(.) > getTimestamp($nlrule)]
   for $touchedRule in $touchedLayer3rules
   return $rules/nlrule[getTimestamp(.) =
   getTimestamp($touchedRule)
   and $touchedRule = getLayer3Rules(.)]
```

[0039] This query presumes that a natural-language security-definition has caused a layer-L3 security-definition to be updated, if both security-definitions have the same time-stamp. In other scenarios, it may be feasible to e.g. introduce a transaction ID to make the determination more precise. The above query furthermore uses a function *getLayer3Rules().* This exemplary function computes all layer-L3 security-definitions for a natural-language security-definition as demonstrated in the first query.

[0040] In summary, the three queries depicted above retrieve, for a given natural-language security-definition, the affected natural-language security-definitions, which have manipulated the same system-specific security-definitions and thus allow for an efficient analysis of the overall security set of the software system.

Claims

1. A method for generating one or more system-specific security-definitions (310, 311, 320, 321) for multiple heterogeneous subsystems (S1, S2) of a software system to enforce the security of the software system, the method comprising the following steps:

   a. storing, in a rule-repository, one or more natural-language security-definitions (100) in XML syntax reflecting a global security intend for the software system which affects the multiple heterogeneous subsystems (S1, S2);
   b. storing, in the rule-repository, one or more machine-readable security-definitions (200) defined in XACML and a first mapping of each natural-language security-definition (100) onto one or more of the machine-readable security-definitions (200);
   c. automatically generating the one or more system-specific security-definitions (310, 311, 320, 321) in XML syntax from the one or more machine-readable security-definitions (200) by one or more rule-converters (RC_1, RC_2) corresponding to the one or more heterogeneous subsystems (S1, S2), wherein the generating is based on a second mapping of each machine-readable security-definition (200) onto one or more of the system-specific security-definitions (310, 311, 320, 321); **characterized in**
   d. selecting at least one of the natural-language security-definitions (100);
   e. retrieving the one or more affected system-specific security-definitions (310, 311, 320, 321) according to the first and second mapping;
   f. storing one or more new system-specific security-definitions (310, 311, 320, 321) in the rule-repository; and
   g. automatically generating one or more machine-readable security-definitions (200) from the one or more new system-specific security-definitions (310, 311, 320, 321) by the one or more rule-converters (RC1, RC2) according to the second mapping.

2. The method of claim 1, further comprising the steps of:

   - selecting at least one of the system-specific security-definitions (310, 311, 320, 321);
   - retrieving the one or more affected natural-language security-definitions (100) according to the first and second mapping.

3. The method of any of the preceding claims, further comprising the steps of:

   - updating the one or more natural-language security-definitions (100) and / or the one or more machine-readable security-definitions (200);
   - updating the corresponding system-specific security-definitions (310, 311, 320, 321) by the one or more rule-converters (RC1, RC2).

4. The method of any of the preceding claims, further comprising the steps of:

   - updating one or more of the affected system-specific security-definitions (310, 311, 320, 321);
   - selecting the one or more affected system-specific security-definitions (310, 311, 320, 321) which have been updated;
   - retrieving the at least one natural-language security-definition (100) which has updated the selected system-specific security-definitions (310, 311, 320, 321).

5. The method of any of the preceding claims, wherein the one or more natural-language security-definitions (100), the one or more machine-readable security-definitions (200) and / or the one or more system-specific security-definitions (310, 311, 320, 321) comprise a time-stamp.

6. The method of any of the preceding claims, wherein the one or more machine-readable security-definitions (200) comprise any of the group of constructs comprising users, roles, resources and / or access-privileges.

7. A rule-repository for generating one or more system-specific security-definitions (310, 311, 320, 321) for one or more heterogeneous subsystems (S1, S2) of a software system, the rule-repository comprising one or more rule-converters (RC1, RC2), the rule-repository being adapted for performing a method of any of the claims 1 - 6.

8. The rule-repository of claim 7, wherein the one or more rule-converters (RC1, RC2) are adapted for generating, for each construct of the one or more machine-readable security-definitions (200) at least one corresponding construct

specific to the one or more system-specific security-definitions (310, 311, 320, 321).

9. The rule-repository of claim 7 or 8, wherein the first and / or the second mapping is stored in the one or more natural-language security-definitions (100) and / or the one or more machine-readable security-definitions (200).

10. The rule-repository of claim 7 or 8, wherein the first and / or the second mapping is stored in at least one separate mapping document.

11. The rule-repository of claim 7-10, wherein the one or more natural-language security-definitions (100), the one or more machine-readable security-definitions (200) and / or the one or more system-specific security-definitions (310, 311, 320, 321) are XML documents.

12. A computer program comprising instructions for performing a method of any of the preceding claims 1 - 6.

Patentansprüche

1. Ein Verfahren zum Generieren einer oder mehrerer systemspezifischer Sicherheitsdefinitionen (310, 311, 320, 321) für mehrere heterogene Subsysteme (S1, S2) eines Softwaresystems zum Durchsetzen der Sicherheit des Softwaresystems, wobei das Verfahren die folgenden Schritte aufweist:

a. Speichern einer oder mehrerer natürlichsprachlicher Sicherheitsdefinitionen (100) in XML Syntax, welche eine globale Sicherheitsabsicht für das Softwaresystem wiedergeben, welches die mehreren heterogenen Subsysteme (S1, S2) betrifft in einem Regelspeicher;
b. Speichern einer oder mehrerer in XACML definierter maschinenlesbarer Sicherheitsdefinitionen (200) und einer ersten Abbildung von jeder natürlichsprachlichen Sicherheitsdefinition (100) auf eine oder mehrere der maschinenlesbaren Sicherheitsdefinitionen (200) in dem Regelspeicher;
c. Automatisches Generieren der einen oder der mehreren systemspezifischen Sicherheitsdefinition (310, 311, 320, 321) in XML Syntax aus der einen oder den mehreren maschinenlesbaren Sicherheitsdefinitionen (200) durch einen oder mehrere Regelkonvertierer (RC1, RC2) entsprechend zu dem einen oder den mehreren heterogenen Subsystemen (S1, S2), wobei das Generieren auf einer zweiten Abbildung von jeder maschinenlesbaren Sicherheitsdefinition (200) auf eine oder mehrere der systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) basiert, **gekennzeichnet durch**:
d. Auswählen zumindest einer der natürlichsprachlichen Sicherheitsdefinitionen (100);
e. Abrufen der einen oder der mehreren betroffenen systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) gemäß der ersten und der zweiten Abbildung;
f. Speichern einer oder mehrerer neuer systemspezifischer Sicherheitsdefinitionen (310, 311, 320, 321) in dem Regelspeicher; und
g. Automatisches Generieren einer oder mehrerer maschinenlesbarer Sicherheitsdefinitionen (200) von der einen oder der mehreren neuen systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) durch den einen oder die mehreren Regelkonvertierer (RC1, RC2) gemäß der zweiten Abbildung.

2. Das Verfahren gemäß Anspruch 1, weiterhin die Schritte aufweisend:

- Auswählen zumindest einer der systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321);
- Abrufen der einen oder der mehreren betroffenen natürlichsprachlichen Sicherheitsdefinitionen (100) gemäß der ersten und der zweiten Abbildung.

3. Das Verfahren gemäß einem der vorhergehenden Ansprüche, weiterhin die Schritte aufweisend:

- Updaten der einen oder der mehreren natürlichsprachlichen Sicherheitsdefinitionen (100) und / oder der einen oder mehreren maschinenlesbaren Sicherheitsdefinitionen (200);
- Updaten der entsprechenden systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) durch den einen oder die mehreren Regelkonvertierer (RC1, RC2).

4. Das Verfahren gemäß einem der vorhergehenden Ansprüche, weiterhin die Schritte aufweisend:

- Updaten einer oder mehrerer der betroffenen systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321);

- Auswählen der einen oder der mehreren betroffenen systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321), welche geupdated wurden;
- Abrufen der zumindest einen natürlichsprachlichen Sicherheitsdefinition (100), die die ausgewählten systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) geupdated hat.

5. Das Verfahren gemäß einem der vorhergehenden Ansprüche, wobei die eine oder die mehreren natürlichsprachlichen Sicherheitsdefinitionen (100), die eine oder die mehreren maschinenlesbaren Sicherheitsdefinitionen (200) und / oder die eine oder die mehreren systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) einen Zeitstempel aufweisen.

6. Das Verfahren gemäß einem der vorhergehenden Ansprüche, wobei die eine oder die mehreren maschinenlesbaren Sicherheitsdefinitionen (200) eines aufweisen aus der Gruppe von Benutzern, Rollen, Ressourcen und / oder Zugriffsprivilegien.

7. Ein Regelspeicher zum Generieren einer oder mehrerer systemspezifischer Sicherheitsdefinitionen (310, 311, 320, 321) für ein oder mehrere heterogene Subsysteme (S1, S2) eines Softwaresystems, wobei der Regelspeicher einen oder mehrere Regelkonvertierer (RC1, RC2) aufweist und der Regelspeicher ausgebildet ist zum Ausführen eines Verfahrens gemäß einem der Ansprüche 1 bis 6.

8. Der Regelspeicher gemäß Anspruch 7, wobei der eine oder die mehreren Regelkonvertierer (RC1, RC2) ausgebildet sind zum Generieren mindestens eines entsprechenden Konstrukts, welches spezifisch für die eine oder die mehreren systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) ist, für jedes Konstrukt der einen oder der mehreren maschinenlesbaren Sicherheitsdefinitionen (200).

9. Der Regelspeicher gemäß Anspruch 7 oder 8, wobei die erste und / oder die zweite Abbildung in der einen oder den mehreren natürlichsprachlichen Sicherheitsdefinitionen (100) und / oder der einen oder den mehreren maschinenlesbaren Sicherheitsdefinitionen (200) gespeichert ist.

10. Der Regelspeicher gemäß Anspruch 7 oder 8, wobei die erste und / oder die zweite Abbildung in mindestens einem separaten Abbildungsdokument gespeichert ist.

11. Der Regelspeicher gemäß der Ansprüche 7 bis 10, wobei die eine oder die mehreren natürlichsprachlichen Sicherheitsdefinitionen (100), die eine oder die mehreren maschinenlesbaren Sicherheitsdefinitionen (200) und / oder die eine oder die mehreren systemspezifischen Sicherheitsdefinitionen (310, 311, 320, 321) XML Dokumente sind.

12. Ein Computerprogramm, welches Instruktionen aufweist zum Ausführen eines Verfahrens gemäß einem der vorhergehende Ansprüche 1 bis 6.

Revendications

1. Procédé pour générer une ou plusieurs définitions de sécurité spécifiques au système (310, 311, 320, 321) pour de multiples sous-systèmes hétérogènes (S1, S2) d'un système logiciel afin de sécuriser le système logiciel, le procédé comprenant les étapes suivantes :

a. stockage, dans un référentiel de règles, d'une ou plusieurs définitions de sécurité en langage naturel (100) en syntaxe XML reflétant une intention de sécurité globale pour le système logiciel qui affecte les multiples sous-systèmes hétérogènes (S1, S2) ;
b. stockage, dans le référentiel de règles, d'une ou plusieurs définitions de sécurité lisibles par machine (200) définies en XACML et d'un premier mappage de chaque définition de sécurité en langage naturel (100) avec une ou plusieurs définitions de sécurité lisibles par machine (200) ;
c. génération automatique des une ou plusieurs définitions de sécurité spécifiques au système (310, 311, 320, 321) en syntaxe XML à partir des une ou plusieurs définitions de sécurité lisibles par machine (200) par un ou plusieurs convertisseurs de règles (RC1, RC2) correspondant aux un ou plusieurs sous-systèmes hétérogènes (S1, S2), dans lequel la génération est basée sur un second mappage de chaque définition de sécurité lisible par machine (200) avec une ou plusieurs des définitions de sécurité spécifiques au système (310, 311, 320, 321) ; **caractérisé par** les étapes de :
d. sélection d'au moins une des définitions de sécurité en langage naturel (100) ;

e. récupération des une ou plusieurs définitions de sécurité spécifiques au système affectées (310, 311, 320, 321) en fonction des premier et second mappages ;
f. stockage d'une ou plusieurs nouvelles définitions de sécurité spécifiques au système (310, 311, 320, 321) dans le référentiel de règles ; et
g. génération automatique d'une ou plusieurs définitions de sécurité lisibles par machine (200) à partir des une ou plusieurs nouvelles définitions de sécurité spécifiques au système (310, 311, 320, 321) par les un ou plusieurs convertisseurs de règles (RC1, RC2) en fonction du second mappage.

2. Procédé selon la revendication 1, comprenant en outre les étapes de :

- sélection d'au moins une des définitions de sécurité spécifiques au système (310, 311, 320, 321) ;
- récupération des une ou plusieurs définitions de sécurité en langage naturel affectées (100) en fonction des premier et second mappages.

3. Procédé selon l'une quelconque des revendications précédentes, comprenant en outre les étapes de :

- actualisation des une ou plusieurs définitions de sécurité en langage naturel (100) et/ou des une ou plusieurs définitions de sécurité lisibles par machine (200) ;
- actualisation des définitions de sécurité spécifiques au système correspondantes (310, 311, 320, 321) par les un ou plusieurs convertisseurs de règles (RC1, RC2).

4. Procédé selon l'une quelconque des revendications précédentes, comprenant en outre les étapes de :

- actualisation d'une ou plusieurs des définitions de sécurité spécifiques au système affectées (310, 311, 320, 321) ;
- sélection des une ou plusieurs définitions de sécurité spécifiques au système affectées (310, 311, 320, 321) qui ont été actualisées ;
- récupération de l'au moins une définition de sécurité en langage naturel (100) qui a actualisé les définitions de sécurité spécifiques au système sélectionnées (310, 311, 320, 321).

5. Procédé selon l'une quelconque des revendications précédentes, dans lequel les une ou plusieurs définitions de sécurité en langage naturel (100), les une ou plusieurs définitions de sécurité lisibles par machine (200) et/ou les une ou plusieurs définitions de sécurité spécifiques au système (310, 311, 320, 321) comprennent une estampille temporelle.

6. Procédé selon l'une quelconque des revendications précédentes, dans lequel les une ou plusieurs définitions de sécurité lisibles par machine (200) comprennent une quelconque construction du groupe de constructions comprenant des utilisateurs, des rôles, des ressources et/ou des privilèges d'accès.

7. Référentiel de règles pour générer une ou plusieurs définitions de sécurité spécifiques au système (310, 311, 320, 321) pour un ou plusieurs sous-systèmes hétérogènes (S1, S2) d'un système logiciel, le référentiel de règles comprenant un ou plusieurs convertisseurs de règles (RC1, RC2), le référentiel de règles étant conçu pour mettre en oeuvre un procédé selon l'une quelconque des revendications 1 à 6.

8. Référentiel de règles selon la revendication 7, dans lequel les un ou plusieurs convertisseurs de règles (RC1, RC2) sont conçus pour générer, pour chaque construction des une ou plusieurs définitions de sécurité lisibles par machine (200), au moins une construction correspondante spécifique aux une ou plusieurs définitions de sécurité spécifiques au système (310, 311, 320, 321).

9. Référentiel de règles selon la revendication 7 ou 8, dans lequel le premier et/ou le second mappage sont stockés dans les une ou plusieurs définitions de sécurité en langage naturel (100) et/ou dans les une ou plusieurs définitions de sécurité lisibles par machine (200).

10. Référentiel de règles selon la revendication 7 ou 8, dans lequel le premier et/ou le second mappage sont stockés dans au moins un document de mappage séparé.

11. Référentiel de règles selon l'une quelconque des revendications 7 à 10, dans lequel les une ou plusieurs définitions de sécurité en langage naturel (100), les une ou plusieurs définitions de sécurité lisibles par machine (200) et/ou

les une ou plusieurs définitions de sécurité spécifiques au système (310, 311, 320, 321) sont des documents XML.

12. Programme informatique comprenant des instructions pour mettre en oeuvre un procédé selon l'une quelconque des revendications 1 à 6.

Fig. 1

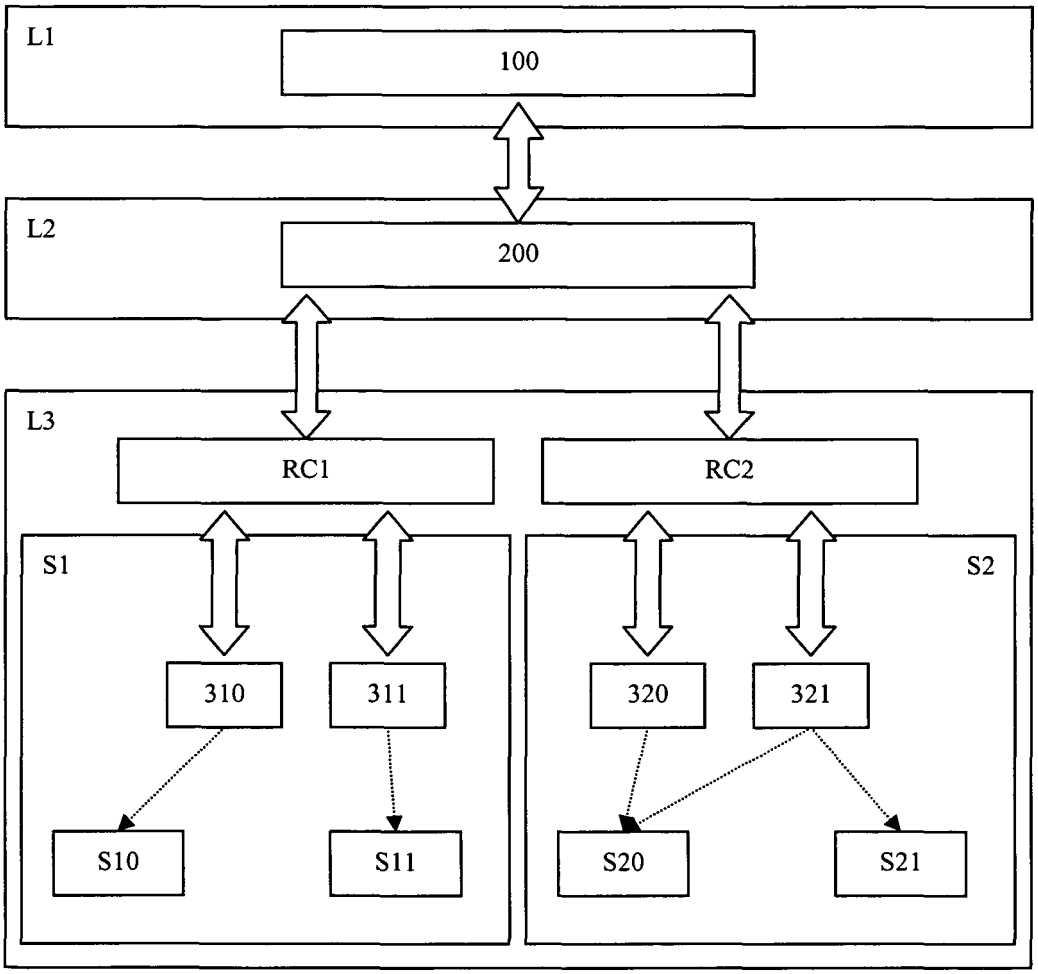

Fig. 2

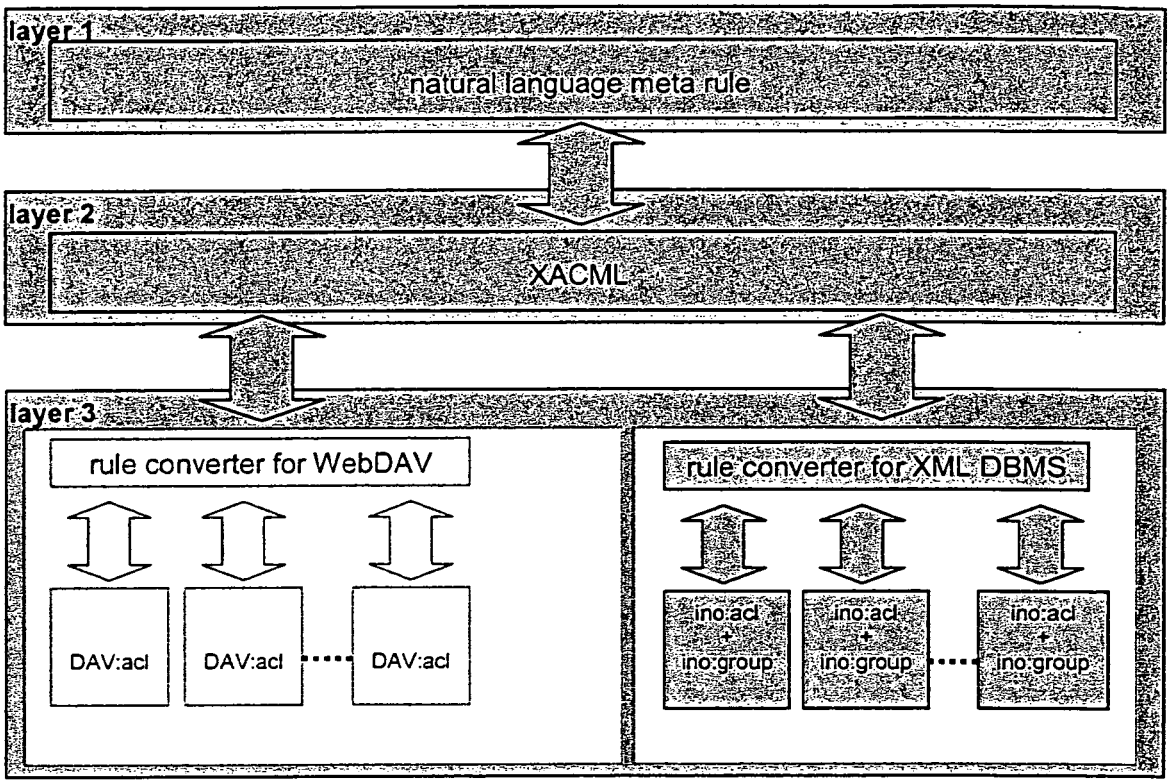

Fig. 3

```
<nlrule ruleId="DevelopersReadRule">
 <rule>Grant read access to all developers on the data relating to the
  project SYLT.</rule>
 <layermapping layer2Id=" DevelopersReadPolicy">
  <layer3mapping type="WebDAV-acl" id="DevelopersReadPolicyAcl1" />
  <layer3mapping type="Tamino-acl" id="developersreadsylt" />
  <layer3mapping type="Tamino-group" id="DevelopersGroup" />
 </layermapping>
</nlrule>
```

Fig. 4

```
<Policy PolicyId="DevelopersReadPolicy"...>
  <Rule ... Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch ...>
            <AttributeValue ...>Developers</AttributeValue>
            <SubjectAttributeDesignator ... />
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch ...>
          <AttributeValue ...>SYLT</AttributeValue>
          <ResourceAttributeDesignator ... />
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch ...>
            <AttributeValue ...>read</AttributeValue>
            <ActionAttributeDesignator .../>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

Fig. 5

```
<D:acl xmlns:D="DAV:" id="DevelopersReadPolicyAcl1">
 <D:ace>
  <D:principal>
   <D:href>http://webdavserver/groups/Developers</D:href>
  </D:principal>
  <D:grant>
   <D:privilege><D:read></D:privilege>
  </D:grant>
  </D:ace>
</D:acl>
```

Fig. 6a

```
<ino:acl ino:aclname="developersreadsylt" xmlns:ino:="...">
 <ino:ace ino:access="read">SYLT</ino:ace>
</ino:acl>
```

Fig. 6b

```
<ino:group ino:groupname="DevelopersGroup" xmlns:ino:="...">
 <ino:aclref>developersreadsylt</ino:aclref>
</ino:acl>
```

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 7058715 B **[0004]**
- US 6751509 B **[0004]**
- US 6978379 B1 **[0005]**
- EP 1308823 A2 **[0006]**
- WO 2008103725 A1 **[0006]**